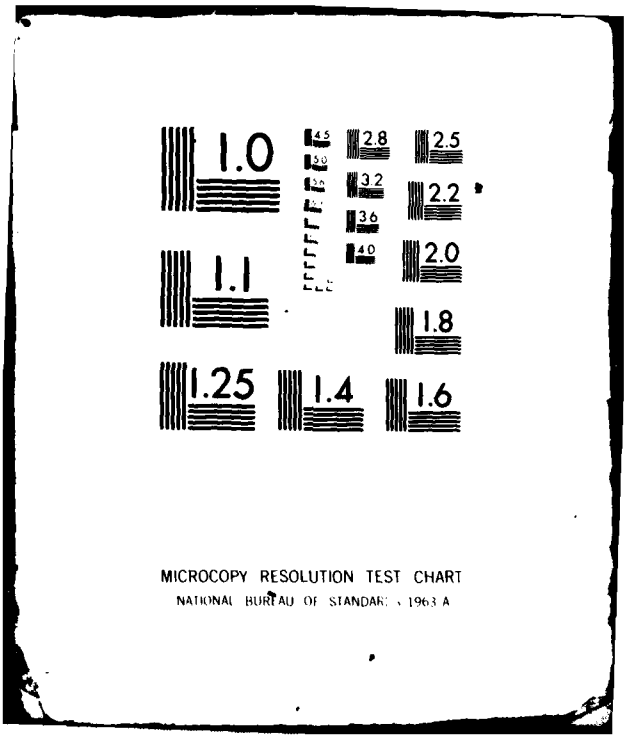
AD-A098 439 UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES DEPT OF--ETC F/G 9/2
MODULAR DESIGN OF OPERATING SYSTEMS USING ABSTRACT DATA TYPES.(U)
JUN 80 P B HANSEN, J FELLOWS DAAG29-77-G-0192

UNCLASSIFIED ARO-15037.2-EL NL

END
DATE FILMED
5-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ARO 15037.2-EL

AD A098439

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BLANK	2. GOVT ACCESSION NO.: BLANK AD-A098439
3. RECIPIENT'S CATALOG NUMBER: BLANK	
4. TITLE (and Subtitle): MODULAR DESIGN OF OPERATING SYSTEMS ~~UNIGN~~ USING ABSTRACT DATA TYPES	5. TYPE OF REPORT & PERIOD COVERED: FINAL REPORT
	6. PERFORMING ORG. REPORT NUMBER: BLANK
7. AUTHOR(s): PER BRINCH HANSEN AND JON FELLOWS	8. CONTRACT OR GRANT NUMBER(s): D AAG29-77-G-0192 NEW
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Department 41063 University of Southern California Los Angeles, California 90007	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: P-15037-M
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709	12. REPORT DATE: June 1980
	13. NUMBER OF PAGES: 44
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): BLANK	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DTIC ELECTE S MAY 5 1981 D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

D

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Trio. Concurrent Pascal. Modular programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the Trio operating system which enables users to simultaneously develop and execute programs at three terminals. The system is written in Concurrent and Sequential Pacal and has been used on a PDP 11/55 mini computer since Spring 1979.

The Trio System is not available for distribution.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

81 5 01 084

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(18) ARO (19) 15037.2-EL

(11) Jun 80

(6) MODULAR DESIGN OF OPERATING SYSTEMS
USING ABSTRACT DATA TYPES.

(12) 10

10 Per Brinch/Hansen
~~and~~
Jonathan/Fellows

(9) Final rept.

U.S. ARMY RESEARCH OFFICE

(15) Contract No. DAAG29-77-G-0192

Computer Science Department
University of Southern California
Los Angeles, California 90007

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

This research tested recent ideas on the use of abstract data types in concurrent programming by implementing a multiterminal operating system for a minicomputer. This system, named Trio, was implemented in Concurrent Pascal, a language which makes it possible to build a large concurrent program out of small modules that can be programmed and tested one at a time. Trio has been used by a team of graduate students for over one year. It has proven to be a system that a single person can understand fully, depend on for months without failure, and adapt easily to changing requirements. These desirable properties of large programs are the most fundamental objectives of all work in programming methodology.

The research had three *major goals:*

(1) To test the simplicity that may result from using abstract data types in the design of realistic operating systems.

Trio suprised its implementors by being simpler than its single user predecessor (Solo).

(2) To test how reliability can be improved by checking access rights to data structures during compilation and by testing operations on data structures one at a time.

During the development of Trio, only one error was discovered that was not caught by compile time checks and module tests.

(3) To set standards for the specification, design, and documentation of non-trivial concurrent programs.

The following publications contain a complete description of the design of Trio:

Brinch Hansen, P. and Fellows, J., The Trio Operating System, Software - Practice and Experience 8, xxxxxxxxx

Fellows, J., The Trio User Manual. Computer Science Department, University of Southern California, Los Angeles, CA, June 1980.

Fellows, J., Applications of Abstract Data Types - the Trio Operating System. Computer Science Department, University of Southern California, Los Angeles, CA, (In preparation).

The authors of this report were the only scientific personnel who participated in this research.

SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 10, 943–948 (1980)

The Trio Operating System*

PER BRINCH HANSEN AND JON FELLOWS
Computer Science Department, University of Southern California, Los Angeles, California 90007, U.S.A.

SUMMARY

This paper is an overview of the Trio system which enables users to simultaneously develop and execute programs at three terminals. The system consists of an operating system written in Concurrent Pascal and a set of standard programs written in Sequential Pascal. The system has been used on a PDP 11/55 minicomputer since spring 1979. This work concludes 5 years of experience with the first abstract language for concurrent programming.

KEY WORDS Concurrent Pascal Operating system Trio

BACKGROUND

This paper is an overview of the Trio operating system which enables users to simultaneously develop and execute programs at three terminals. The system consists of an operating system written in Concurrent Pascal and a set of standard programs written in Sequential Pascal. The system has been used on the PDP 11/55 minicomputer since spring 1979.

This work is a continuation of earlier work that led to the development and implementation of the programming language Concurrent Pascal which includes processes, monitors and classes.[1]

The focus of this research has been the concept of an abstract data type—the combination of a data structure and all the possible operations on it into a single program module. This concept contributes to simplicity by locating details of data representation and transformation in modules instead of spreading them throughout a large program. It increases reliability by making it possible for a compiler to prevent data structures from being destroyed by arbitrary operations.

So far Concurrent Pascal has been used to design a single-user operating system, a job stream system, a real-time scheduler, and a message passing system for a multicomputer network.[2–4] It has also been used to build several microcomputer operating systems.

Each of these model systems has the following characteristics:

1. Each concurrent program consists of modules of less than one page each.
2. Each module consists of a data structure and a set of procedures which can be called by other modules. These procedures provide the only means of changing

* This research was supported by the Army Research Office under Contract No. DAAG-29-77-G-0192.

0038–0644/80/1110–0943$01.00

Received 21 July 1980

the data. This protection of data integrity is enforced during compilation only and is not supported by run-time mechanisms.

3. Each module can only call the procedures defined within a small number of other modules. The access rights of modules to the procedures of other modules are also checked during compilation.
4. The modules are connected hierarchically to one another. So a module cannot call itself indirectly through other modules. This too is verified by the compiler.
5. The modules were tested one at a time from the bottom towards the top (but may, of course, be conceived top down). The compilation checks mentioned above ensure that new (untested) modules do not make old (tested) modules fail.
6. Each of these programs was built and described in complete detail by a single programmer in a matter of weeks.

The resulting systems have been more reliable than the hardware they run on.

The aim of the Trio system is to demonstrate the practicality of using the same programming concepts to build an operating system of medium size. The following is only an overview of the function and structure of Trio. The reader is referred to the user manual for more detail.[5]

The Trio system is not currently available for distribution.

SYSTEM OPERATION

The Trio system is built for a PDP 11/55 minicomputer with 80 K words of store, a disk drive (of 1 M words), a multiplexor with three alphanumeric display terminals, a magnetic tape drive and a line printer.

The Trio system permits three programmers to use the minicomputer simultaneously. The three users are assumed to be members of a (possibly larger) programming team which is developing a set of related programs.

The programs of a user team are stored as text and code files on a removable disk pack. The files that are used by the whole team are described in a single system catalog on the disk, while those that are still being developed are described in one or more user catalogs. The system does not restrict a user to operate on the files of a single user catalog. Multiple user catalogs are provided to enable users to group their files into convenient subclasses and to operate on different subclasses simultaneously.

The system is operated by the users themselves. Each session typically lasts an hour or more. At the beginning of the session, the user team mounts its own disk pack and starts the Trio system. Each user then sits down at one of the terminals and types a command that gives him access to the files described in a given user catalog as well as the files described in the system catalog.

A user can now input, compile, edit and test Pascal programs. When a program is finished, the user can move its description from the given user catalog to any other catalog (including the system catalog).

The users can make copies of text files on the line printer. But the system makes display and editing of text at the terminals so convenient that the need for printed listings is reduced considerably compared to the Solo system.

It is possible to copy the files of a single catalog (or the whole disk) onto magnetic tape and use it to re-establish disk files after hardware or software failure.

The execution of a program can be pre-empted by depressing the bell key on a user terminal. The system then displays the line number of the last statement that was executed in the program. This is a very convenient mechanism for locating endless loops in new programs while they are being tested.

USER PROCESSES

In the single-user operating system Solo, concurrency could only be exploited by performing the input, execution and output of a single program simultaneously. In the Trio system, simultaneity is achieved by the much simpler means of executing three user processes simultaneously. No attempt is made to perform the input/output and program execution of a single-user process concurrently.

The concurrent program Trio resides permanently in the main store together with three user processes of fixed size. Each user process serves a single user terminal.

A user process executes a cyclical Pascal program called 'do' which accepts commands from the corresponding terminal. Each command specifies the execution of a Pascal program with a set of arguments of type identifier, integer or boolean. The given program is loaded from the disk and executed as a procedure called by the do program. Any Pascal program can call any other Pascal program stored on the disk by means of a standard procedure 'run' implemented by the operating system.

The operating system maintains a set of booleans which represent the resources shared by the user processes. These resources are primarily the line printer, the magnetic tape and the disk. A standard procedure enables a user program to request and release exclusive access to any subset of these resources. Within the operating system these resources are acquired one by one in fixed order to prevent deadlock.[6] When a user program is pre-empted by depressing the bell key, the operating system automatically releases all resources requested but not yet released by the given user process.

The processor is treated as a composite resource which consists of three processor shares—one for each user process. When a user types a command to the do program the latter requests exclusive use of the corresponding processor share before executing the program named by the command. When that program terminates or fails, the do program temporarily releases the processor share again. When a user wishes to execute a concurrent program he needs exclusive access to the whole machine. This is achieved by requesting the use of all three processor shares—an action that delays the execution of the concurrent program until the other user processes have completed their current execution of programs.

The interface between the Trio system and any of its user programs is a set of procedures that are implemented by the operating system and are called by the user programs. The names and parameter types of these procedures are defined by a piece of text, called the prefix, which is put in front of every user program before it is compiled.

These procedures give each user program simultaneous access to at most four sequential files. Two of these files are text files that are read and written character by character. The others are files that are input and output in blocks called disk pages.

Other procedures enable programs to use the terminals, to create and delete files on the disk, and to call other programs stored on the disk. We have tried to make this interface much more convenient to the programmer than the interface of the Solo system.

FILE SYSTEM

The file system is the most critical part of the operating system. Not only is it the long-term storage of the users, but it also resides on a mechanical device that is several orders of magnitude slower and much less reliable than the computer itself.

The Trio file system is an extension of the Solo file system.

To avoid occasional, but time-consuming relocation of data on the disk, the pages allocated to a single file are addressed indirectly through a page map—a single disk page which defines the addresses of the data pages of the file. The page map allows the operating system to place the data pages anywhere on the disk and let them remain there until the file is deleted.

The system catalog is a file that starts at a fixed disk address and describes the name and attributes of all common files (including itself). The attributes of a file are the disk address of its page map, its protection status (true or false), and its kind (scratch, text, code or catalog).

Each user catalog is described in the system catalog as a file of type 'catalog'. At the beginning of a terminal session, all file names used within a user process are looked up in the system catalog. A user can now select a given user catalog by name. Following this, all file names used by the corresponding process are first looked up in the given user catalog, and (if that fails) they are then looked up in the system catalog. The set of catalogs that are used by a process at any given time is known as its current catalog set.

A file is opened by looking it up in the current catalog set and bringing its page map into the main store. The file is looked up by converting its name to a hash key that defines the starting point of a cyclical search in one of the catalogs.

Since each user team has its own removable disk pack with a separate user catalog for each programmer (or subtask), files need only be protected against accidental overwriting and deletion. All files are initially unprotected. The user protects a file by calling a standard program which sets the protection boolean in the file attributes to true.

The Trio file system then is a hierarchical system with three levels. The first level is the system catalog which describes common files and user catalogs. The second level consists of the user catalogs which describe disjoint classes of user files. And the third level consists of the user files. Descriptions of files can be moved freely from any catalog to any other catalog. The protection facilities are minimal and do not prevent users from operating simultaneously on the same files in a time-dependent manner. Although this philosophy is adequate for program development from a small number of terminals, it is not secure enough for a general time-sharing system.

We would like to emphasize that since small operating systems become inexpensive when they are written in abstract programming languages, one can afford to tailor each of them for a single purpose. We have followed this approach in designing an operating system that is convenient for a computer with three user terminals. But we have made no attempts to make the same design applicable to larger systems with five or ten terminals.

SIZE AND PERFORMANCE

The Trio system consists of a Concurrent Pascal program and a set of standard programs of the following lengths:

Trio program	1,600 lines
Do program	500 —
File program	900 —
Edit program	900 —
Catalog programs	2,700 —
Device programs	1,900 —
Other programs	700 —
New programs	9,200 lines
Compilers	17,000 —
Total system	26,200 lines

The design of the system was started in the spring of 1978 by the authors. The initial implementation was done over a period of 1 year by a graduate student (Jon Fellows) who was unfamiliar with Concurrent Pascal to begin with. During 1979 the system was used experimentally and tuned. We estimate that a full-time programmer who knows Concurrent Pascal could have done it in 6–8 months. By comparison Solo was developed in 3 months.

The compilers for Concurrent and Sequential Pascal were moved from the Solo system with minimal changes.

Trio requires a main store of 80 K words which is used as follows:

Concurrent Pascal kernel	4 K words
Operating system	7 K words
User processes	69 K words
Main store	80 K words

A text file can be created or deleted in 1–2 s depending on its length (< 128 K char). It can be opened in 240 ms and then read or written at the rate of 1000 char/s. The overall performance of the system for non-trivial processing can best be illustrated by the compilation time which is 13 s + 3 ms/char for a single user. When two users are compiling simultaneously, the compilation time for each of them becomes 23 s + 5 ms/char. The compilation time reaches 35 s + 7 ms/char when *all three* users are compiling simultaneously (a very unlikely situation). This means that the operating system itself (of 1600 lines) can be recompiled in 2–5 min.

These execution times are primarily limited by the speed of code interpretation and to a lesser extent by the slow disk. The compilers generate abstract code which is interpreted by the well-known technique of 'threaded code'.

FINAL REMARKS

In a recent book[3] one of us said this:

> 'The operating systems written so far in Concurrent Pascal are small. I would hope (and expect) that a larger system will turn out to be 'more of the same.' But it seems worthwhile to confirm this by using Concurrent Pascal to build a medium-size operating system, for example, a terminal system that gives each user the capability of Solo.'

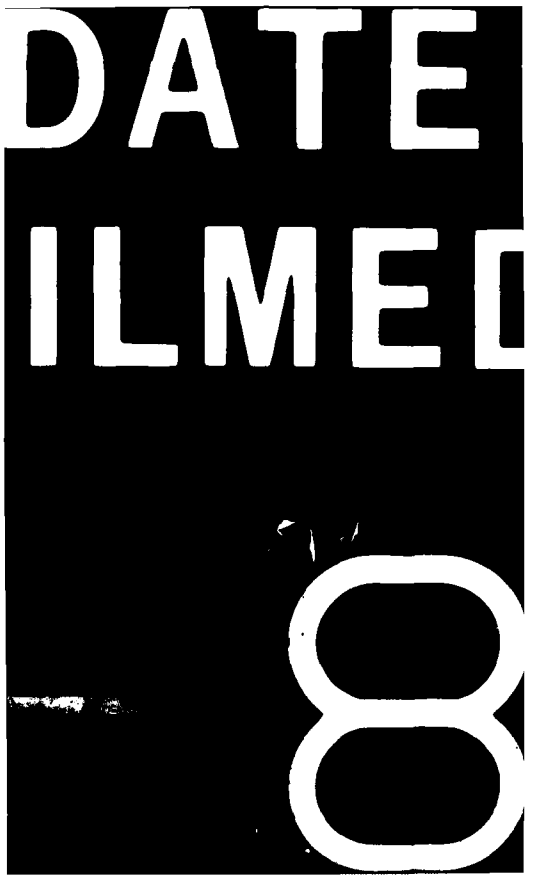
Well, we have done it now and it was more of the same. The real surprise was that Trio in many ways turned out to be simpler than Solo! This concludes 5 years of experience with the first abstract language for concurrent programming. The underlying concepts of processes, monitors and classes can now be regarded as proven tools for software engineering. So it is time to do something else.

ACKNOWLEDGEMENT

Habib Maghami derived a single-user version of Trio known as the Mono system. This work has been supported by the Army Research Office under contract DAAG–29–77–G–0192.

REFERENCES

1. P. Brinch Hansen, 'The programming language Concurrent Pascal', *IEEE Trans. on Software Engineering*, 1 (2), 199–207 (1975).
2. P. Brinch Hansen, 'The Solo operating system', *Software—Practice and Experience*, 6 (2) 141–205 (1976).
3. P. Brinch Hansen, *The Architecture of Concurrent Programs*, Prentice-Hall, Englewood Cliffs, N.J. 1977.
4. P. Brinch Hansen, 'Network: a multiprocessor program', *IEEE Trans. on Software Engineering*, 4 (3) (1978).
5. J. A. Fellows, *The Trio User Manual*, Computer Science Department, University of Southern California, Los Angeles, CA., 1980.
6. P. Brinch Hansen, *Operating System Principles*, Prentice-Hall, Englewood Cliffs, N.J. 1973.


DATE
ILMED
8